(12)

Bolt Beranek and Newman Inc.

AD-A112994

Report No. 4928

Development of a Voice Funnel System

Quarterly Technical Report No. 14
1 November 1981 to 31 January 1982

March 1982

DTIC
ELECTE
APR 6 1982
S D
H

Prepared for:
Defense Advanced Research Projects Agency

DTIC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

82 04 05 087

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD A112 994	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Development of a Voice Funnel System Quarterly Technical Report No. 14		5. TYPE OF REPORT & PERIOD COVERED Quarterly Technical 14 1 November 81-31 January 81
		6. PERFORMING ORG. REPORT NUMBER 4928
7. AUTHOR(s) R. D. Rettberg		8. CONTRACT OR GRANT NUMBER(s) MDA903-78-C-0356
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA Order No. 3653
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd., Arlington, VA 22209		12. REPORT DATE March 1982
		13. NUMBER OF PAGES 38
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

DTIC ELECTE APR 6 1982 D

H

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Voice Funnel, Digitized Speech, Packet Switching, Butterfly Switch, Multiprocessor

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

↳This

This Quarterly Technical Report covers work performed during the period noted on the development of a high-speed interface, called a Voice Funnel, between digitized speech streams and a packet-switching communications network.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 4928 Bolt Beranek and Newman Inc.

DEVELOPMENT OF A VOICE FUNNEL SYSTEM

QUARTERLY TECHNICAL REPORT NO. 14
1 November 1981 to 31 January 1982

March 1982

This research was sponsored by the
Defense Advanced Research Projects
Agency under ARPA Order No.: 3653
Contract No.: MDA903-78-C-0356
Monitored by DARPA/IPTO
Effective date of contract: 1 September 1978
Contract expiration date: 31 December 1981
Principal investigator: R. D. Rettberg

Prepared for:

Dr. Robert E. Kahn, Director
Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, VA 22209

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either express or implied, of the Defense Advanced Research Projects Agency or the United States Government.

Table of Contents

Accession For
NTIS GRA&I ✓
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

DTIC COPY INSPECTED 3

FIGURES

1. Introduction

This Quarterly Technical Report, Number 14, describes aspects of our work performed under Contract No. MDA903-78-C-0356 during the period from 1 November 1981 to 31 January 1982. This is the fourteenth in a series of Quarterly Technical Reports on the design of a packet speech concentrator, the Voice Funnel.

The design of the Voice Funnel application software (as distinguished from the Butterfly Multiprocessor hardware and the Chrysalis operating system) has been described previously in BBN Report Number 4098, "Design of a Voice Funnel System". Although the basic structure laid out there is in fact quite close to that of the final system, its details have undergone numerous changes and revisions since the writing of that report. This report reviews the progress of the implementation effort to date, and gives an overview description of the final design.

2. Progress to Date

The development of the Voice Funnel software has recently had two important demonstrations. In July, the application was demonstrated in a rudimentary form, as speech packets were successfully looped through the Butterfly from one Lexnet Packet Voice Terminal (PVT) to another. In December, we shipped the first Funnel to Lincoln Laboratory and in January demonstrated the first packet speech over the Wideband Network using the Funnel.

Through November of 1981, the application software effort was concerned primarily with developing a "stand-alone" version of the Voice Funnel application program. This version uses many of the basic primitive operations available under the Chrysalis operating system, but not its more sophisticated features.

Since November, the focus of the application software development effort has broadened to cover several new activities, including: (1) testing the stand-alone Funnel in the Wideband Network; (2) adapting the stand-alone Funnel to Chrysalis; (3) development of software modules to support the routing of ST packets; (4) adapting the Funnel to the new addressing conventions currently being introduced into the Wideband Network; and (5) documenting and reviewing the application software design. We have also been giving some attention to the longer term issues of performance testing and remote monitoring and control. The remainder of this section reviews these activities in more detail.

2.1 Lexnet Interface

The first operational software to be demonstrated on the Butterfly was a partial implementation of the Voice Funnel Application that was capable of interfacing to a Lexnet through a Lexnet Concentrator Interface (LCI), and routing speech packets from one Packet Voice Terminal to another. This was important not so much for the application software, which was fairly simple, but for the microcode, application, and operating system functions that were in place and operating reliably.

In developing the Lexnet interface software, we were forced to deal with one unexpected problem associated with the LCI. Since the microprocessor that controls the synchronous interface between the Butterfly and the LCI is not fast enough to service the transmit and receive channels simultaneously, it must inhibit interrupts from the receiver DMA hardware while it is transmitting a packet to the Butterfly. Thus, when the LCI enters transmit mode, the Butterfly is not allowed to send more than one packet before the LCI leaves that mode.

The result is that the LCI behaves as a half duplex interface to the Lexnet. Under a steady two way traffic load, the number of packets that the Butterfly sends to the LCI cannot exceed the number of packets that the LCI sends to the Butterfly. Lincoln Laboratory has solved this problem by modifying the LCI design to support a full duplex synchronous interface. Currently, only one improved LCI has been built. We hope that more will be available by the time the number of Packet Voice Terminals on the Wideband Net is large enough to require the higher bandwidth of a full duplex LCI.

For the interim, we have used the packet scheduling facility provided by the Butterfly I/O hardware to guarantee that packets will not be dropped at this interface. By scheduling packets for transmission at an interval that slightly exceeds the time that it takes to transmit a maximum length packet, we guarantee that no more than one packet will ever arrive while the LCI is in transmit mode.

Since the size of the average packet produced by all Lexnet PVTs currently used in Voice Funnel experiments is close to the maximum Lexnet packet size, the extra inefficiencies introduced by this scheme are small. If packets from the Lexnet begin to vary significantly in size (e.g. when new vocoders are introduced) the inefficiencies may become intolerable.

We should note that the UMC-Z80 used as a high speed synchronous interface by the Lincoln Mini-Concentrator and by the PDP-11/44 used for packet video experiments at ISI has the same problem.

2.2 Stand-alone Funnel

Once it was shown that there were sufficient operating system resources in place to support the development of the Voice Funnel, the operating system and application development efforts diverged. The then existing version of Chrysalis was frozen and used to provide a stable environment for further Voice Funnel development while development of Chrysalis continued separately.

By the end of October, we had finished debugging the HDLC to VDH converter that connects the Funnel to the PSAT at the signal level, we had tested our implementation of the PSAT Host Access Protocol, and we had expanded our Internet software. At this point, the development was delayed until after the Thanksgiving Holiday, as the Lexnet hardware that we needed for testing was tied up by a Wideband Network demonstration.

During the first two weeks of December, we tested the Funnel with a Lexnet, a PSAT, and a satellite simulator in the back room at BBN. On December 15, we were able to loop full duplex speech traffic between a pair of Packet Voice Terminals through the Satellite Simulator.

2.3 Wideband Network Testing

In the first week of December, we had also delivered a ten processor Butterfly to Lincoln Laboratory, and established communication between that machine and the Lincoln PSAT. The next logical step was to see whether the stand-alone Funnel could send speech traffic over a real satellite channel.

Due to a series of operational problems with the Wideband Network, this was a time-consuming but not very productive effort. Our first opportunity to test with an operational network was on January 18, when the Lincoln site was capable of carrying speech traffic. At that time, we repeated the experiment that we had done at BBN, using the Lincoln PSAT, the Lincoln Voice Funnel, and the satellite channel in place of the analogous components at BBN. In addition to other tests, a continuous full duplex speech connection through the satellite channel was maintained for thirty minutes without incident. A two node network was not available to us until February third. At that time, a solid connection between the Lincoln Funnel and the ISI Mini-Concentrator was established, and cross country speech was transmitted through the Funnel for the first time.

2.4 Chrysalis Adaptation

In December, an updated version of Chrysalis was released and the two development efforts began to converge. The new Chrysalis version incorporates all of the major facilities called for in the original design and replaces all of the ad hoc mechanisms used to support the Funnel during the summer.

Adapting the Voice Funnel to this new version of Chrysalis involves two major subtasks. First, the support software that the Funnel uses for functions related to the Synchronous I/O system, I/O buffer management, and interprocess communication constitutes some of the earliest software to run on a Butterfly processor node. During December, January, and February we redesigned these modules to fit the new environment and to correct several deficiencies that we had observed.

The second subtask is the adaptation of the software that implements the "higher level" components of the Funnel, such as those that deal with packet routing and PSAT status monitoring. Since these are more dependent on protocol specifications than they are on the immediate programming environment, they should remain substantially unchanged.

2.5 Wideband Network Addressing

At the end of January, the PSAT group released Wideband Note Number 36, a description of a new set of Host Access Protocol (HAP) addressing conventions. These conventions are designed to bring the PSAT subnet into conformance with the new addressing scheme outlined in Wideband Note Number 27. In February, a new version of the stand-alone Funnel, compatible with the "new style" HAP addressing, was developed and tested with the PSAT at BBN. Until this change is installed in the Wideband Network, we will use compiler switches to stay compatible with both addressing environments. Further modifications to the addressing conventions used in the Funnel will be made when the Lexnet PVTs are brought into conformance with the new Wideband Net Addressing conventions.

2.6 Documentation

Since the completion of the stand-alone version of the Funnel, we have been working on a formal update to the design documentation for the Funnel. This update will appear in two installments, corresponding to the two major subtasks described above. The first installment, documenting the higher level design of the Funnel, appears in Section 3 of this report. The second installment, describing support software for the Funnel application, is under review and will appear in a future report.

2.7 Remote Monitoring and Control

While the development of a robust, operational Voice Funnel is our first goal, remote monitoring and control is clearly an essential facility if the Funnel is to be of long term use. We have explored this problem in a preliminary way, with the intent of defining and implementing facilities compatible with the NU monitoring and control system already used with the PSAT subnet. What we have discovered is that new protocols for host monitoring and control have recently been defined. Integration of the new Host Monitoring protocol into the NU system is currently underway, and support for hosts using this protocol will be available in the near future.

3. Voice Funnel Application

In this section, we describe the design of the Voice Funnel application software. For background purposes, we start with an overview of the task that the Funnel is meant to perform, a summary of the Internet addressing conventions that the Funnel follows when it is routing packets to Wideband Network hosts, and a brief discussion of the programming environment. We assume that the reader is generally familiar with the Internet Stream protocol (ST), an extension of the DOD Standard Internet Protocol designed to support the transmission of real time data. A detailed specification of this protocol is given in Internet Experiment Note 119. We also assume some familiarity with the operation of the Wideband Network. For a more detailed discussion of Wideband Network issues pertinent to the operation of the Voice Funnel, see the "PSAT Technical Report" (BBN Report No. 4469).

3.1 Voice Funnel Overview

The primary task of the Voice Funnel is to interface a large number of speech terminals, multiplex their data streams for delivery over the Wideband Satellite Network, and demultiplex these data streams for delivery to terminals at other sites. Speech packets consist of transport and speech protocol information plus speech data. The position of the Funnel in a typical Wideband Network site is shown in Figure 1. Data is passed to and from the Wideband Network through the Pluribus Satellite IMP (PSAT), while a collection of hosts gain access to the Wideband Network through the Voice Funnel via direct connections or through local networks.

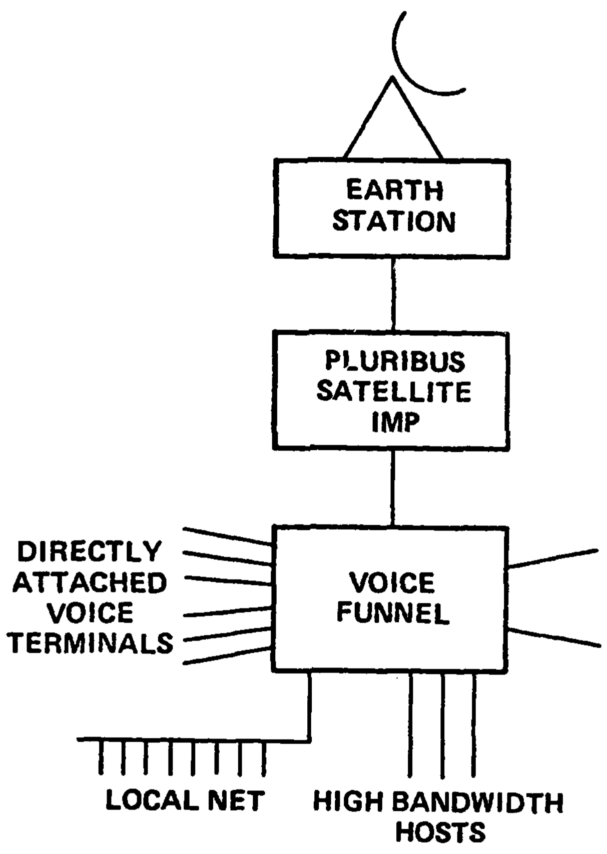

Figure 1 . ARPA Wideband Network Site

In its role of transport and multiplexing, the Voice Funnel is not limited to traffic from packet voice terminals. Any communications resource that uses protocols recognized by the Funnel may be attached to it. Examples include conventional hosts, or new applications that wish to take advantage of the high bandwidth offered by a satellite channel to transmit such things as high resolution graphics or video images. In this report we focus on speech, since that is the initial application.

The Voice Funnel must support two kinds of data traffic, corresponding to the two modes of operation offered by the Wideband Network. The simplest and most familiar of the two is the datagram. The treatment of datagram traffic in the Wideband Network conforms to accepted notions, and need not be discussed at length, except to note that it takes a relatively long period of time for a datagram to traverse the satellite channel. Since the scheduling algorithm used on the channel requires that space be reserved before a datagram can be transmitted, it takes a minimum of two 270 millisecond round trip delays to send a datagram.

The delay and scheduling overhead associated with datagram transmission is unacceptable for a high duty cycle, low delay application such as packet speech. To support this and other applications with similar requirements, the Wideband Network offers a stream mode of transmission. In this mode, a Voice Funnel requests a dedicated allocation of channel bandwidth, called a stream. If it is available, the stream is assigned to the Funnel that requested it and is freed for use by others only when it is relinquished by that Funnel. Once the stream has been established, there is no need to reserve space on a per message basis, so stream packets do not have to wait an extra round trip delay the way datagrams do. In addition, the stream mechanism guarantees that there will be sufficient network resources to support a call from beginning to end, unless the stream is preempted by higher priority traffic. With datagram traffic, network resources are requested on a per packet basis with no long term guarantee.

For point-to-point speech, each Voice Funnel on the Wideband Network maintains a single stream. Each Funnel multiplexes all of its output speech traffic into this stream. The Funnels use the "flow specification" information included in ST call setup request packets to expand their streams when a call is initiated. The flow specification information is also retained and used later on to contract the stream when a call is terminated.

3.2 Wideband Network Addressing

The Voice Funnel makes its routing decisions according to the Wideband Net addressing scheme agreed upon at the Wideband Network meeting at Lincoln Laboratory in May 1981. The Internet Address format adopted at that time is shown in Figure 2. The "Net Number" field of the address is governed by the conventions established by the DOD standard Internet Protocol. For hosts on the Wideband Network, its value is always 28 (decimal). The "Reserved" field has been reserved for future expansion, and is currently ignored in all packet routing decisions. The "HAP Number" and "Local" fields are used to route packets within the Wideband Network.

```
 0          7 8          15 16         23 24          31
+------------+-------------+------------+-------------+
| Net Number | HAP Number  |  Reserved  |    Local    |
+------------+-------------+------------+-------------+
```

Figure 2 . Internet Address Format for Wideband Net

With certain restrictions, the interpretation of the HAP Number and Local fields is quite simple. The HAP Number field identifies hosts that are "directly connected" to the Wideband Network through PSAT host interfaces. Voice Funnels and Mini-Concentrators are examples of hosts that are assigned HAP Numbers. The local field distinguishes among hosts that are connected to the Wideband Network through the same directly connected host. Lexnet Packet Voice Terminals are examples of such hosts.

```
 0          7 8          15
+------------+-------------+
| HAP Number |    Local    |
+------------+-------------+
```

Figure 3 . HAP Address Format for Wideband Net

The routing of packets within the Wideband Network has been simplified by defining the HAP address of a host to be the concatenation of the HAP Number and Local fields in its Internet address, as shown in Figure 3. The assignment of values in the HAP Number field to existing hosts has been specified in Wideband Network Note Number 30, and is summarized in Figure 4. The interpretation of the Local address field is site dependent.

HAP Number	Usage
00-07:	127 PSATS with 16 internal hosts each plus 16 "global" hosts.
08-0F:	8 Gateways (currently internal to the PSATS).
10-F7:	232 directly connected hosts on the Wideband Network.
F8-FF:	2048 Group Addresses.

Figure 4 . Current HAP Number Assignments

As an example, a typical routing scenario is illustrated in Figure 5. The following steps are performed in processing a packet:

1. An Internet packet arrives at host A, which is directly connected to the Wideband Network. The packet is destined for host C, which is connected to the Wideband Net through host B.

2. Host A notices that the packet is headed for a destination somewhere on the Wideband Net, but that the HAP number field does not match its own. It therefore forms a HAP destination address by concatenating the

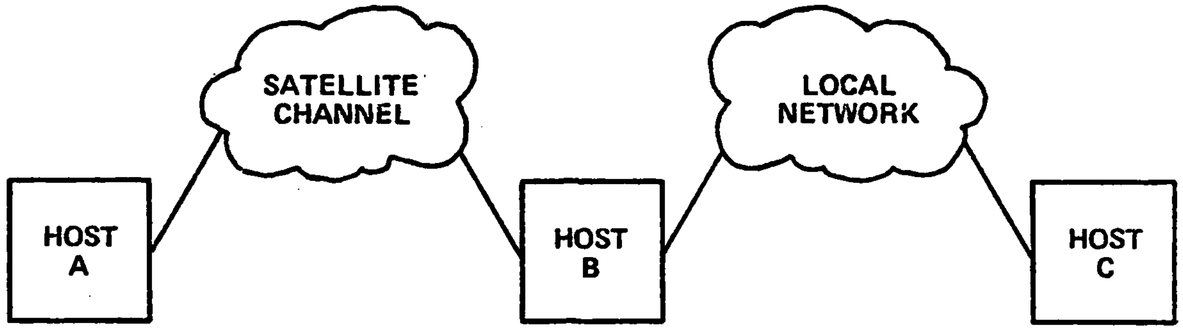

Figure 5 . Wideband Network Routing Scenario

HAP Number and Local Fields of the Internet Destination address, and submits the packet to the local PSAT.

3. The packet travels through the Wideband Net to Host B. Note that the source and destination PSATs need use only the high order byte of the HAP destination address to route the packet.

4. Host B receives the packet. Since both the Network Number and HAP Number fields match its own, Host B knows that Host C must be local to it. Referring to site specific information, Host B routes the packet to host C.

3.3 Programming Environment

The Voice Funnel application relies on Chrysalis for its programming environment. This operating system, developed specifically for the Butterfly Multiprocessor, has been described in a general way in BBN Report Number 4098. More detailed descriptions will appear in future reports. This section reviews certain general aspects of Chrysalis in order to clarify the discussion of the application software.

One of the most important characteristics of Chrysalis is that it encourages and enforces process oriented software. The process is the basic program unit and the basic unit of parallelism. The structure of the Voice Funnel application is therefore based on cooperating processes.

Chrysalis provides four major functions. First, it provides a scheduling service. It keeps track of which processes are runnable and which are not, and gives each runnable process its share of CPU time as dictated by parameters supplied when the process was created. Second, it serves as a resource manager. It creates and deletes processes, and allocates memory space on demand. Third, it allows convenient access to primitive operations supported by the Processor Node Controller, through a subroutine package referred to here and in other documents as the "Chrysalis Protected Library". Finally, it provides debugging and error handling facilities.

Another important characteristic of Chrysalis is its local orientation. With the exception of one node that supports a small number of centralized facilities, every processor runs the same operating system software. This means that each node manages resources that are local to it. Allocation of resources on a remote node is handled by sending a request to the operating system on that node.

The net result of the local, process-based orientation of Chrysalis is a programming environment that is surprisingly similar to that of an ordinary time sharing system on a single processor. In particular, there are no special language constructs for parallel execution at the statement or subroutine level. The interprocess communication mechanisms, on the other hand, are quite sophisticated compared to those that appear in most single processor systems.

3.4 The Critical Path

We have used three strategies for decomposing the activity in the Voice Funnel into parallel processes. First, we have isolated the critical path in the Voice Funnel and implemented it as a pipeline. Second, we have removed all functions that do not require real time service from the critical path and placed them in separate processes. Finally, we have created multiple processes to share the workload at each stage of the critical path. In this section we talk about the structure of the critical path in the Voice Funnel, and how we introduce parallelism into that path to increase performance.

Figure 6 presents an elementary view of the pipeline that implements the critical path in the Voice Funnel. Note that this diagram represents the bulk of the processing activity in the Voice Funnel, but only a fraction of its functionality. Activities such as call setup, satellite stream setup, and monitoring and control all happen outside of this pipeline. In the following subsections, we discuss the specific functions of each stage in the critical path. Since the Input and Output stages are similar in structure, we discuss these before we present the Routing and Aggregation stage. We describe the Task Queue, which is a data structure, not a process, in a later section. For the purposes of this discussion, it is convenient to speak of Input and Output as if each channel were serviced by two different processes. In fact, the structure of the I/O hardware is such that it is more convenient to combine these functions into a single process for each channel.

3.4.1 Input

The first stage of the pipe is an input process that services the I/O hardware, deals with local header information for each incoming packet, and puts each packet on a queue for

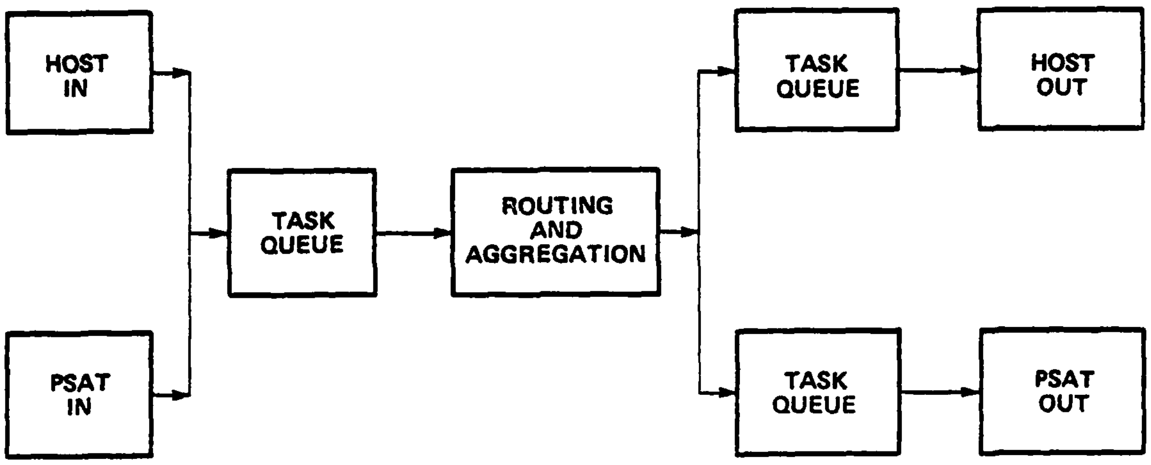

Figure 6 . Voice Funnel Critical Path

further processing. Currently, there are two kinds of input process. For each physical link to the PSAT, there is a HAP (PSAT Host Access Protocol) input process. For each physical link to a Lexnet, there is a Lexnet input process. More input process types will be added as new types of hosts are connected to the Funnel.

The duties of these processes can be classified as being specific to the Interface Hardware or specific to the protocol used to communicate across the link. The hardware specific duties are common to all input process types. They include recycling empty receiver buffers as they are returned, checking input packets for error flags set by the hardware, and dealing with fatal hardware error conditions. The specific nature of these duties and the methods by which they are carried out will be the subject of a future report.

The protocol specific duties of a Lexnet input process are quite simple, since there is no exchange of status or control packets between the Lexnet and the Funnel at this level. The input process simply checks the destination address of each incoming packet for correctness and places properly addressed packets on a Task Queue for the Routing and Aggregation Process.

The protocol specific duties of a HAP input process are somewhat more complicated. This process uses the HAP header to decide whether an incoming packet is a status packet, a setup packet to be handled by the HAP setup process, or an ordinary data packet. If it is a data packet, the input process verifies the header checksum and (if the checksum is valid) passes the remainder of the packet to the Routing and Aggregation Process. Otherwise, the packet is passed to a HAP status process or a HAP setup process, where it is checksummed and dealt with appropriately.

3.4.2 Output

The final stage of the pipe is a process that is responsible for last minute packet processing and for setting up packets for output. Like the input processes, there are currently two types of output process, with more to be added as new types of host and network are supported by the Funnel. Also like the input processes, output processes have hardware specific duties that are common to every output process, and protocol specific duties that vary among process types.

Hardware specific duties include freeing transmit buffers after they have been processed by the I/O hardware and dealing with fatal hardware error conditions. The specific nature of these duties and the methods by which they are carried out will be the subject of a future report.

The protocol specific duties of the Lexnet output processes amount to transferring each outgoing packet from a task queue to an output queue that is interpreted by the I/O hardware. The protocol specific duties of the HAP output processes are somewhat more complicated. Processing a typical data packet for output requires the following operations:

1. Remove the packet from a task queue.
2. Insert a sequence number into the HAP header and adjust the header checksum to account for it.
3. Enqueue the packet to the I/O hardware.

The scheduling of stream packets for transmission to the PSAT at proper intervals is not handled at this level. Instead, it is handled by a Stream Slot Scheduler Process which takes care of setting up aggregation buffers and passing them to the PSAT Output Processes. This is facilitated by the Butterfly I/O hardware, which allows the application software to specify an absolute transmission time for any outgoing packet by storing a value into a data structure associated with the packet buffers.

3.4.3 Routing and Aggregation

The middle stage of the pipe is a Routing and Aggregation Process that uses Internet and ST header information to decide how each data packet that passes through the Funnel should be handled. ST packets are transferred to aggregation buffers, which are subsequently passed to the PSAT output process at appropriate times by the Stream Slot Scheduler Process. For Internet Packets, the Routing and Aggregation Process adds the local header, transfers the packet to an output node, and enqueues the packet directly to the appropriate output process. The specific set of operations performed on each packet is as follows:

1. Block transfer the header to a scratch area in local memory and checksum it.
2. Process the options field.
3. Use the header contents to decide where the packet should go.
4. If it is an Internet Packet, block transfer it to the node that is connected to the destination network and build the local header for the packet.
5. If it is an ST packet, block transfer it to an aggregation buffer.
6. Free the buffer space on the input processor node.

7. If it is an Internet Packet, put it on a task queue for the appropriate output process.

Note that for Internet packets, this process really operates at two different protocol levels. At the Internet level, it makes a routing decision. At the local network level, it constructs a header. This incorporation of two functions into a single process is purely an efficiency measure. It reduces context switching and interprocess communication overhead by reducing the number of separate processes that must deal with each packet to the minimum required for the introduction of effective parallelism. We have been careful to structure this process at the subroutine level in such a way as to make it easy to add new types of local networks.

The information derived from the routing decision in step three includes the protocol used on the destination network, the address of the destination host on the destination network, the identity of the Task Queue for the Output process, and the identifier of a buffer pool on the output node. This completely specifies the operations to be performed in the remaining steps. The method by which this information is obtained depends on the packet type.

For ST packets, the connection ID field of the packet is used as an index into a routing table which contains all of the above information, plus the connection ID for the next hop. The ST routing table is maintained by a separate process.

For Internet Packets, the decision is a little more complicated. If the network number field of the destination address does not correspond to the Wideband Network, a routing table is used to retrieve pertinent information about the next hop. For hosts on the Wideband network, either the "local address" field of the destination address is used as an index into a lookup table, or the "HAP number" and "local address" fields are concatenated to form a HAP destination address.

The block transfer operation is needed for packets whose source and destination networks are connected to different processor nodes. This is because the I/O hardware can read data only from memory that is on the processor node that is local to it. Note that the Routing and Aggregation Process does not need to be on either the source or the destination processor node to initiate a block transfer between them.

Most of the packets that arrive from the PSAT will actually be ST envelopes that contain many smaller packets, destined for several different Hosts. For this kind of packet, the Routing and Aggregation process will execute a series of routing decisions and a series of block transfers, one for each packet in the envelope.

Conversely, most of the packets that traverse the satellite channel will be packed into envelopes and sent as stream packets rather than datagrams. When the Routing and Aggregation process gets this kind of packet, it appends it to an "aggregation area", using a block transfer operation. The Routing and Aggregation Process is assisted by a separate process, not shown in Figure 6, that takes care of setting up aggregation areas and scheduling the transmission of stream frames to the PSAT. The data structure that implements the aggregation area is associated with a lock so that the assistant process and multiple instances of the Routing and Aggregation Process can manipulate it without conflict.

3.4.4 Adding Parallelism to the Critical Path

The Task Queue is the key to introducing parallelism into the segments of the Voice Funnel critical path. This mechanism relies on the dual queue primitives provided by the Processor Node Controller. These primitives are described in Quarterly Technical Report No. 12. The important characteristics of the Task Queue mechanism are:

1. Atomic Operations

   Since insertion and removal of data items is supported by indivisible microcoded primitives, a Task Queue can have multiple readers and writers without locks.

2. Speed Advantage

   The microcoded primitives that support these operations are fast. Insertion and removal operations execute in approximately ten microseconds, with an added time penalty of only 5 microseconds if the queue is on a remote node. If one adds overhead for error checking and a library subroutine call, the total execution time is still less than 100 microseconds. A comparable operation on the Pluribus takes approximately half a millisecond.

3. Wait Operation

When a Task Queue is empty, reader processes can use it to store Event Handles, which are posted when new tasks arrive. This allows the reader processes to go to sleep until there is something to do, instead of consuming resources by continuously polling the queue.

Given the properties of the Task Queue, we can introduce the expanded view of the Voice Funnel critical path shown in Figure 7. Since operations on a Task Queue are indivisible, multiple readers and writers are possible. Furthermore, the speed of these primitives keeps the Task Queues from being bottlenecks. This simplifies the structure of the application software by allowing it to define a common collection point for similar tasks. Note that it is also possible for a single process to service multiple task queues.

Since both Task Queues and processes are software entities that can be dynamically created and deleted by the Operating System at the request of the application software, an arbitrary number of processes and Task Queues can be active on the Butterfly at any one time. In particular, it is possible for an arbitrary number of processes to share the Routing and Aggregation workload. The practical limit on the number of instances of the Routing and Aggregation process is the number of available Processor Nodes. Similarly, we can dedicate a single process to each I/O channel, spreading the processing load as evenly as the I/O hardware configuration permits.

Note that the association between processes and Processor Nodes is very flexible. It is determined by constraints such as load leveling, the I/O hardware configuration, and minimizing switch contention.

Figure 7 also illustrates another important aspect of Task Queues: the identities of the readers need not be known to the writers. This has an important consequence for the Funnel-to-PSAT interface. By having a pair of physical interfaces implemented on separate nodes, we achieve added reliability with very little overhead, since the loss of one of the output nodes will only degrade throughput. It cannot hang the system by leaving the task queue locked, for instance.

Note that detection of, and reaction to, Processor Node failure is a function of higher level application processes, along with the Operating System. It is not part of the Task Queue mechanism. The methods for discovering that a particular host is no longer operational are specific to the protocol used to access that host. Similarly, it is up to the Routing and Aggregation Processes, and the processes that support them, to keep track of which hosts are active and which are not, and to avoid sending traffic to those that are not.

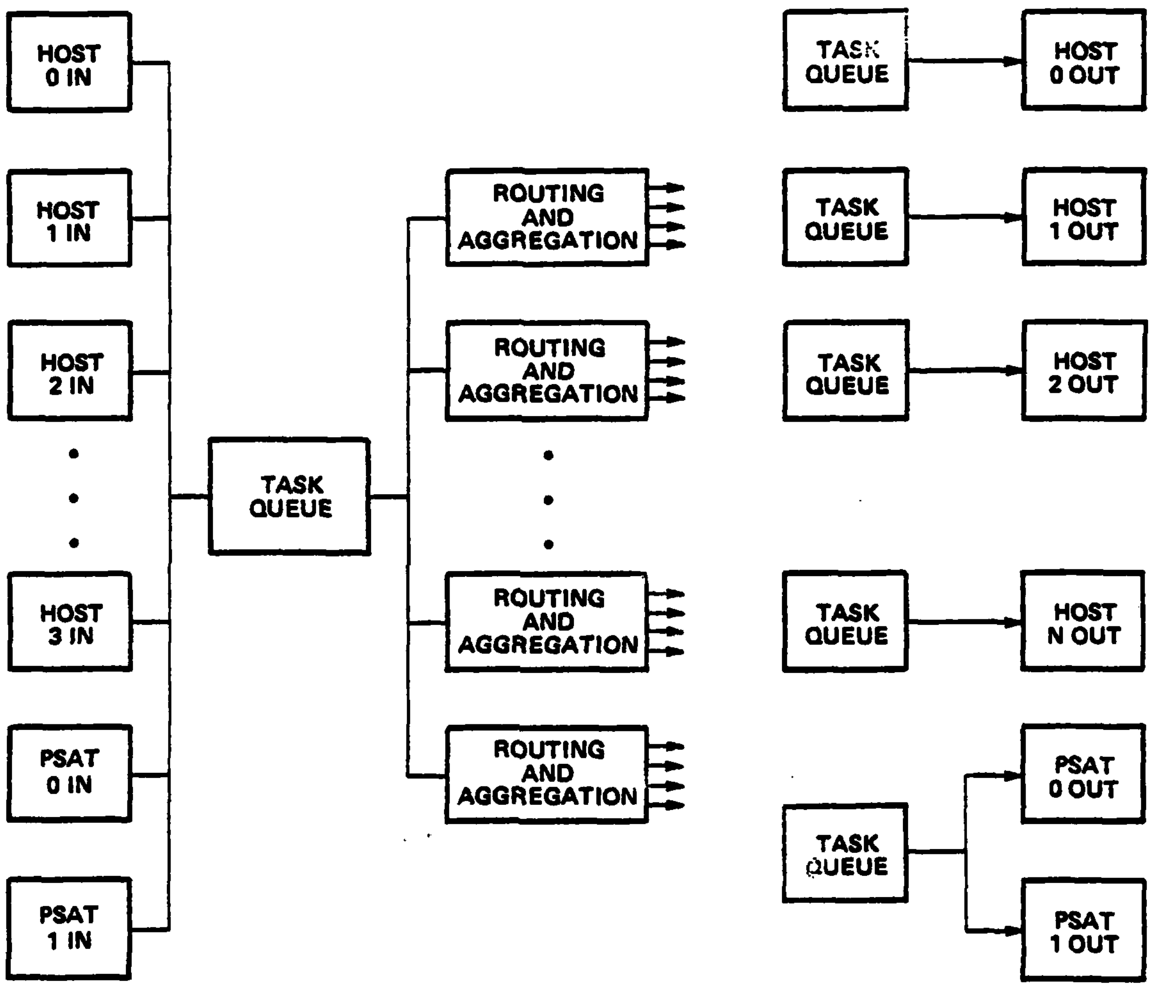

Figure 7 . Voice Funnel Critical Path With Added Parallelism

3.5 Support Processes

So far, we have explained how we speed up the critical path in the Voice Funnel by breaking it into stages and spreading the work load over multiple processes operating on multiple processors. The other strategy that we have mentioned is one that removes from the critical path all processing that is not time critical. This processing load is assumed by a collection of "support processes". In general the response time required of these processes is not as short as that required of the processes on the Critical Path. This does not necessarily mean that these processes will operate any more slowly, however.

The principal method of communication among these processes is similar to that of processes in the Critical Path. Each process services one or more Task Queues, and produces results that are placed on other Task Queues. Some of these processes also communicate implicitly with the Routing and Aggregation Processes by updating ST and Internet routing tables. The paths of communication among these processes are much more varied than those in the Critical Path, as the functions performed tend to be more complicated.

The functions of the support processes in the Voice Funnel are summarized in the following subsections. We have omitted any discussion of how the Funnel handles the Acceptance/Refusal (A/R) information that is part of the PSAT Host Access Protocol specification because this is an optional feature of the protocol and it is not clear that it will be used in its present form.

Since these support processes are expected to be relatively low bandwidth activities, the current implementation of the Funnel has only one instance of each process. Should it turn out that some process is overloaded, we may implement the locking conventions necessary to support multiple instances of that process.

3.5.1 HAP Status

This process generates and receives status packets from the PSAT, as specified in Section 4.1.5 of the PSAT Technical Report. There is one instance of this process for each physical link that is open to the PSAT. If this process does not receive any status packets from the PSAT for a period of ten seconds, or if it receives a Restart Request from the PSAT, it must initiate a link restart exchange, as specified in Section 4.1.6 of the PSAT Technical Report. It does this by posting an event to the input/output process associated with the link. After the input/output process takes care of reinitializing the hardware and exchanging the appropriate sequence of packets with the PSAT, it posts an event back to the HAP Status Process, and normal operation resumes.

3.5.2 HAP Setup

This process handles setup requests to the PSAT, as specified in Section 4.1.4 of the PSAT Technical Report. It in turn receives requests from other processes that need services from the PSAT. In the current Voice Funnel design, this process acts exclusively as an agent for the ST control process, which needs to acquire and release network resources at call setup and take down. However, it is organized in such a way that any process can cause it to request Wideband Network resources.

3.5.3 ST Control

All ST Control Messages are handled by this process. It is therefore the responsibility of this process to implement such things as the call setup procedures that are part of the ST protocol. This involves processing and forwarding call setup requests, updating tables for the Routing and Aggregation process, and asking the HAP setup process to modify stream parameters and establish groups.

3.5.4 Stream Slot Scheduler

The Stream Slot Scheduler acts as an assistant to the Routing and Aggregation Processes. Its purpose is to ensure that stream packets arrive at the PSAT at the proper time. For each active stream belonging to the Voice Funnel, the Stream Slot Scheduler executes the following series of operations once per frame.

1. Lock the aggregation data structure associated with the stream in question.
2. Set up an aggregation area for the next frame, and set up the aggregation data structure to point to it.
3. Unlock the aggregation data structure, keeping a pointer to the old packet.
4. Fill out the ST envelope header of the old packet.
5. Enqueue the old packet to a HAP I/O driver for transmission to the PSAT.

In general, there may be several Routing and Aggregation Processes waiting to transfer data into the aggregation area when the time comes to send the stream packet. If the Stream Slot Scheduler simply waited in line for access to the aggregation data structure, it might transfer the packet to the PSAT too late to catch the next stream slot. To avoid this problem, the Stream Slot Scheduler uses a "priority lock" operation, supported by the locking mechanism implemented by the Processor Node Controller, to put itself at the head of the queue of processes waiting for the lock.

Each aggregation area is set up to accept as much data as the associated stream is capable of handling in a single frame. If the amount of data that arrives in a given frame interval exceeds the capacity of a frame, the excess packets will be dropped. This is consistent with the philosophy that it is up to the hosts at either end of an ST connection to decide how much bandwidth to reserve, and to stick to their self-imposed quota. Since voice packets tend to be small, loss of packets due to breakage is not expected to be a serious concern.

Making the decision to discard packets at this level has both a positive and a negative effect. The advantage of this method is that it is much simpler and more efficient than keeping track of the resources used per frame by each ST connection. The disadvantage is that it does not necessarily penalize the offending connection when the traffic load exceeds available capacity. Since we expect that ST hosts will normally stay within the bandwidth quotas that they have set for themselves, we do not think that this problem will be serious.

3.5.5 Logging Process

For performance measurement and debugging purposes, the Stream Slot Scheduler is supplemented with a Logging Process. Each time a Routing and Aggregation process handles an ST data packet, it constructs a 32-bit word containing the connection ID and size of the packet. If the Logging Process is active, the Routing and Aggregation process places this quantity on a Dual Queue that is serviced by the Logging Process. Using the information that arrives on this queue and information supplied by the ST control process, the Logging Process can construct a fairly detailed profile of the speech traffic passing through the Funnel. This process starts and stops its logging activity at the request of the Voice Funnel Controller. Under ordinary circumstances, it will be inactive.

3.5.6 Internet Routing Update

This process is responsible for maintaining the Internet routing tables used by the Routing processes and the ST Control process. In the current Wideband Network environment, it is only used as a convenience measure for interactively modifying Internet routing tables as speech hosts are added and deleted for experimental purposes. When the Wideband Network becomes part of the Internet, this process will participate in the gateway to gateway exchanges that are used to maintain and modify routing information automatically. This process would also participate in any automated procedure that may be developed for adding and removing Wideband Network speech terminals.

3.5.7 Voice Funnel Controller

This is an interactive process that allows someone at a terminal to examine the state of the Voice Funnel and to modify various parameters. Currently, the user interface is through a console terminal on the Butterfly. We plan to upgrade this process to be capable of communicating over a network using Packet Core, XNET, or some other protocol.

3.5.8 Voice Funnel Monitor

This is the process to which all other processes in the Funnel report unusual events, using a mechanism that is similar in intent to the Trap mechanism used in the Pluribus, though quite different in implementation. The Monitor process is responsible for forwarding these messages to the Controller process for display on the console terminal, and to a Network Monitoring Host, if available. This process also acts as the point of contact for all remote monitoring activity.

3.5.9 Echo Host

This process is present mainly for debugging and testing purposes. Like the PSAT Echo Host, this process has its own internet address. For Internet packets, it reverses the Internet source and destination address of any packet that it receives, and returns the packet to its source. We plan to support both ST and Internet traffic through the Echo Host.

3.5.10 Message Generator

This process is controlled interactively through the Voice Funnel Controller. It is capable of sending single messages or continuous streams of messages to any Internet address that the user cares to specify. It is currently used in conjunction with the Message Sink as a probe to determine whether a particular communication path is operating correctly. For instance, a message to the PSAT echo host at a particular site should return if that site is operational and the path to that site is open. We plan to redesign this process in the near future to support performance testing.

3.5.11 Message Sink

This process is present mainly for debugging and testing purposes. Like the Message Sinks in the PSAT, it has its own internet address. For each packet that it receives, it increments a counter and, if desired, passes a copy of the packet header to the Controller for display on the console. We plan to redesign this process in the near future to support performance testing.

DISTRIBUTION OF THIS REPORT

Defense Advanced Research Projects Agency
Dr. Robert E. Kahn (2)
Dr. Vinton Cerf (1)

Defense Supply Service -- Washington
Jane D. Hensley (1)

Defense Documentation Center (12)

USC/ISI
Dr. Danny Cohen (2)

MIT/Lincoln Labs
Dr. Clifford J. Weinstein (3)

SRI International
Earl Craighill (1)

Rome Air Development Center
Neil Marples - RBES (1)
Julian Gitlin - DCLD (1)

Defense Communications Agency
Gino Coviello (1)

Bolt Beranek and Newman Inc.
Library
Library, Canoga Park Office
R. Bressler
R. Brooks
P. Carvey
P. Castleman
W. Edmund
G. Falk
J. Goodhue
S. Groff
E. Hahn
E. Harriman
F. Heart
M. Hoffman
M. Kraley
A. Lake
W. Mann
W. Milliken
M. Nodine
R. Rettberg
P. Santos
E. Starr
E. Wolf